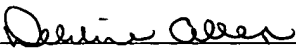


<p style="text-align: center;">CERTIFICATE OF MAILING VIA EXPRESS MAIL 37 C.F.R. §1.10</p> <p>PURSUANT TO 37 C.F.R. 1.10, I HEREBY CERTIFY THAT I HAVE A REASONABLE BASIS FOR BELIEF THAT THIS CORRESPONDENCE IS BEING DEPOSITED WITH THE UNITED STATES POSTAL SERVICE AS EXPRESS MAIL POST OFFICE TO ADDRESSEE ON THE DATE INDICATED BELOW, AND IS ADDRESSED TO:</p> <p style="text-align: center;">MAIL STOP PATENT APPLICATION COMMISSIONER FOR PATENTS P.O. BOX 1450 ALEXANDRIA, VA 22313-1450</p> <p> NAME</p> <p>DATE OF MAILING: NOVEMBER 21, 2003 EXPRESS MAIL LABEL: EV339225688US</p>
--

APPLICATION FOR LETTERS PATENT

FOR

**DEVICE FOR CONTROLLING PROCESSING OF DATA
ELEMENTS OF A DATA STREAM**

This application claims priority to German Application No. 102 54 653.3
filed November 22, 2002

INVENTOR(S): **Lorenzo DiGregorio**
Schleissheimer Str. 91
D-80797 München Germany

Xiaoning Nie
Brehmstr. 10
D-81543 München Germany

Thomas Wahl
Sturmiusstr. 1
D-36037 Fulda Germany

ATTORNEY DOCKET NUMBER: **068758.0146**

CLIENT REFERENCE: **IO387US/MGL/pp**

HOU03:942092.2

DEVICE FOR CONTROLLING PROCESSING OF DATA ELEMENTS OF A DATA STREAM

Priority

5 This application claims foreign priority of the German application
DE 10254653.3 filed on November 22, 2002.

Technical Field of the Invention

 The invention concerns a device laid out to control processing of
elements of a data stream.

10 Background of the Invention

 Data streams, containing data elements that originate from different
data sources, are frequently transmitted in data nets. The data elements of the
different data sources are then arranged sequentially in a data stream. The data
elements are sent to processing in the sequence in which they are present in the data
15 stream.

 During processing of the data elements, execution of very simple,
lightweight instructions can be involved. A stipulated sequence of such instructions is
referred to as a thread. During processing of the data elements of a data stream,
different threads are generally processed. Which thread is processed at which time
20 generally depends on the properties of the data element that is foremost on the data
stream at this time. The choice of thread depends, in particular, on the data source,
from which the data element to be processed originates. For this purpose, data
elements are often provided with an identification number that characterizes their
corresponding data source. The allocated thread can be identified by means of this
25 identification number.

Each thread is assigned its own context. The context contains information concerning the state, in which processing of the thread at the present time is situated. Before the beginning of execution of a thread, the first instruction to be executed in the thread can be entered in the registers allocated to the context. After
5 the beginning of execution of the thread, these registers can be overwritten with the actual instructions of the thread.

A device that assigns the instruction from the corresponding thread, prescribed for an incoming data element, and decodes this instruction is necessary for processing of data elements, so that a component connected after this device, which
10 carries out actual processing of the data element, can be supplied control signals for processing.

Ordinary devices that fulfill the purpose just mentioned are based either on software solutions that run on a digital signal processor, or on hardware solutions. However, software solutions are rather inefficient for byte-byte-processing. Hardware
15 solutions have the drawback of limited flexibility with respect to protocol changes.

Summary of the Invention

The task of the invention is to devise an apparatus for controlling processing of data elements of a data stream that has high efficiency and high flexibility.

20 The task underlying the invention can be solved by a device for controlling processing of data elements, in which a thread is assigned to each data element and no more than one data element enters the device at one time, comprising a first unit, in which the context for each thread is entered, and which fetches an instruction during a first clock cycle that is entered in the context of the thread
25 assigned to the incoming data element, a second unit, which, during a second clock cycle, fetches an instruction, which succeeds a stipulated instruction in the sequence of

instructions of a stipulated thread, and a third unit, which, during the second clock cycle, decodes the instruction that is provided for processing of the data element and fetches a control signal for processing of the data element.

5 The object can also be achieved by a method for controlling processing of data elements, comprising the steps of:

- assigning a thread to each data element and no more than one data element enters the device at one time,
- fetching an instruction in a first unit during a first clock cycle that is entered in the context of the thread assigned to the incoming data element,
- 10 - fetching an instruction in a second unit, which succeeds a stipulated instruction in the sequence of instructions of a stipulated thread, and
- decoding the instruction that is provided for processing of the data element and fetching a control signal for processing of the data element in a third unit.

15

The instruction fetched by the second unit can be the instruction, whose position in the sequence of instructions of the stipulated thread, is the increment of the position of the stipulated instruction. The second unit can be fed with the increment of a count value and an identification value, which designates a thread, and the second
20 unit, by means of the increment and the identification value, may determine the instruction which assumes in the thread designated by the identification value the position designated by the increment assumes. The first unit may activate the context of the thread assigned to the incoming data element, if the preceding data element refers to another thread. The first unit may fetch an instruction of the thread stated in
25 the activated context and transmit this instruction, which is the first instruction of the thread, in particular, to the third unit for decoding, and the first unit may transmit the increment of the position that the instruction fetched by it assumes in the thread, to the second unit. The second unit may determine the instruction that succeeds the

instruction fetched by the first unit in the thread. For data elements entering the device in succession, the same thread can be assigned, as long as the same instruction is used, until a stipulated condition is met. Repetition of an instruction can be accomplished by the fetching of the same control signal by the third unit. The number
5 of repetitions of an instruction can be stipulated by a value, this value, during a repetition of the instruction, can be decremented by the third unit, and the repetitions can be interrupted at the value 0. After fulfillment of the stipulated condition for processing of the data element entering the device next, a stipulated instruction within the thread can be used, if the same thread is assigned to this data element. The inquiry
10 into fulfillment of the stipulated condition may occur in the third unit. The stipulated instruction can be the instruction fetched by the second unit. A connection line for data transmission between the second unit and the third unit can be provided, through which the instruction, fetched by the second unit is transmitted to the third unit. The instruction fetched by the second unit can be transmitted to the first unit and entered in
15 the context there. The stipulated instruction can be fetched by the first unit and transmitted to the third unit for decoding. The third unit, after fulfillment of the stipulated condition, can transmit an instruction to the first unit as to which instruction is to be fetched. The stipulated condition, whose fulfillment leads to interruption of repetitions of an instruction, can be fulfilled by a signal controllable from outside of
20 device, or by a specific data element entering the device, or by a specific state of the corresponding thread, or by a specific instruction to be processed. A program memory can be provided, in which the instructions for processing of the data elements are entered, and in which information is entered for each instruction on how many data elements the instruction is to be applied, wherein the program memory has program
25 lines, in particular, in which one instruction and the corresponding information, with reference to the number of repetitions, are entered. Two series-connected delay units can be provided that delay the data element by one clock cycle each.

The device according to the invention is used to control processing of data elements and includes a first unit, a second unit and a third unit. In the device
HOU03:942092.2

according to the invention, no more than one data element enters at a time. A thread is assigned to each data element.

The context for each thread is entered in the first unit. During a first clock cycle, an instruction is fetched from the first unit, which is entered in the context
5 of the thread assigned to the incoming data element.

The second element fetches an instruction during a second clock cycle that succeeds a stipulated instruction in the sequence of the instructions of a stipulated thread.

The third unit decodes the instruction during the second clock cycle,
10 which is prescribed for processing of the data element, and generates a control signal for processing of the data element.

The instruction prescribed for processing of the data element can be fed from the first or second unit to the third unit in a preceding clock cycle. This instruction can also be determined by means of a stipulated sequence of instructions or
15 based on a condition algorithm.

The control signal generated by the third unit can be conveyed, together with the data element, to a component connected after the device according to the invention, so that actual processing of the data element can be performed in the downline component.

20 The device according to the invention permits control of data element processing in a much more efficient manner than it is possible with a digital signal processor. The device according to the invention also has a high degree of flexibility, since many parameters that contribute to operation of the device can be stipulated.

The device according to the invention operates according to the
25 principle of a data flow machine (data driven). This means that the device according

to the invention only continues processing if a new data element enters the device. In contrast to a data flow machine, a Von Neumann machine operates sequentially with each clock cycle. This is independent of whether a new data element is present or not.

5 The device according to the invention is laid out so that the instruction required for processing of a data element is always present. This is also the case when a new context must be activated, because of a thread change. The device according to the invention therefore makes possible processing of the incoming data elements with avoidance of data congestion (bubble-free). This feature is not exhibited by the known multi-reading machines.

10 The instruction fetched by the second unit is preferably the instruction that immediately succeeds the stipulated instruction in the stipulated thread.

 For the aforementioned purpose, the second element is advantageously supplied with the increment of a count value and an identification value that designates a thread. By means of the increment and identification value, the second
15 unit determines the instruction that assumes the position designated by the increment in the thread designated by the identification value.

 This method of operation of the second unit has the advantage that the instruction that succeeds the instruction just used in the corresponding thread is always fetched by it. Should this instruction be required in the subsequent clock cycle, it is
20 available without delay.

 According to an advantageous embodiment of the invention, the context of the thread, which is assigned to the data element entering the device, is activated by the first unit, if the preceding data element referred to another thread.

 In this case, the first unit preferably fetches an instruction of the thread
25 stated in the activated context. This instruction, which can be the first instruction of the thread, in particular, is conveyed to the third unit for decoding. The increment of

the position that the instruction fetched by it assumes in the thread is sent to the second unit from the first unit.

The second unit advantageously determines the instruction, by means of the obtained increment, that immediately succeeds the instruction fetched by the first unit in the thread.

A particularly preferred embodiment of the invention proposes that, for data elements entering the device in succession, the same thread is assigned, as long as the same instruction is used, until a stipulated condition is met.

During repetition of an instruction, the same control signal can always be generated in a third unit.

In addition, the number of repetitions of an instruction can advantageously be stipulated by a value. This value is decremented during each repetition of the instruction by the third unit. As soon as the value equals zero, the repetitions are interrupted.

Another particularly preferred embodiment of the invention is characterized by the fact that, after fulfillment of the stipulated condition for processing of the data element entering the device next, a stipulated instruction within the thread is used, if the already activated thread is assigned to this data element.

In this case, inquiry into fulfillment of the stipulated condition preferably occurs in the third unit.

The stipulated instruction, to which a jump is made after fulfillment of the stipulated condition, for example, can be the instruction fetched by the second unit.

By a connection line for data transmission between the second unit and the third unit, the instruction fetched by the second unit can preferably be transmitted without delays to the third unit.

It can also be prescribed that the instruction fetched by the second unit is also conveyed to the first unit and entered there in the context.

As an alternative to the instruction fetched by the second unit, after fulfillment of the stipulated condition, a jump can be made to an instruction that is entered in the first unit. This is then conveyed to the third unit for decoding.

In order to select an instruction from the first unit, after fulfillment of the stipulated condition, an instruction is preferably sent to the first unit by the third unit, which states which instruction is to be fetched.

The stipulated condition, whose fulfillment leads to interruption of repetitions of the instruction, can be fulfilled, for example, by a controllable signal from outside of the device, or by a specific data element entering the device, or by a specific state of the corresponding thread, or by a specific instruction being processed.

Another advantageous embodiment of the invention is characterized by a program memory, in which the instructions for processing of the data elements are entered. In addition, information about how many data elements the corresponding instructions are to be applied to is also entered in the program memory for each instruction. The program memory can be arranged in one of the three units of the device according to the invention. In particular, the program memory can have program lines, in each of which an instruction and the corresponding information with reference to the repetition number are entered.

The device according to the invention advantageously comprises two series-connected delay units that delay the data element by one clock cycle. Since the device requires two clock cycles, in order to generate the control signal for processing of the data element, it is guaranteed by the two delay units that the data element and the control signal reach the downline components simultaneously in the device.

Brief Description of the Drawings

The invention is further explained below with reference to the drawing. The drawing shows the single figure of a schematic of a practical example of the device according to the invention.

5 Detailed Description of the Preferred Embodiments

A device 1 is shown in the figure, which has a contact switch unit CS (context switch), an instruction fetch unit IF (instruction fetch), an instruction decoding unit ID (instruction decoding), delay elements D1, D2 and D3, as well as multiplexers MUX1, MUX2 and MUX3.

10 The components just mentioned of device 1 each have inputs and outputs for communication with the other components and for control. These inputs and outputs are described below.

 The context switching unit CS has inputs for a context identification value context_id_i, a state word id_state_s, a context identification word
15 if_context_id_s, an instruction set data element wb_ir_s, a speculative count value id_sp_s and a control signal cs_store_s. On the output side, the context switching unit CS issues a speculative counting value cs_spc_s, an instruction set data element cs_ir_s and a state element cs_state_s.

 The instruction fetch unit IF has two input connections to the outputs of
20 the multiplexer MUX1. At its outputs, the instruction fetch unit IF issues an instruction set data element if_ir_s and a context identification value if_context_id_s.

 The instruction decoding unit ID has two input connections to the outputs of the multiplexer MUX2 and an input connection to an output of the multiplexer MUX1 and the output of the delay element D2. The context identification
25 values if_context_id_s and cs_context_id_s, as well as a useful data element data_s,

also supply the instruction decoding unit ID. At its outputs, the instruction decoding unit ID issues the speculative counting value `id_spc_s`, the control signal `cs_store_s`, a control signal `cs_ir_select_s`, an instruction set data element `id_ir_s`, the state element `id_state_s`, the control signal `if_source_s` and a control signal `dec_o`.

5 The lines for the speculative counting value `id_spc_s`, the context identification value `if_context_id_s`, the speculative counting value `cs_spc_s` and the context identification value `cs_context_id_s` are connected to the inputs of multiplexer MUX1. A control input of multiplexer MUX1 is provided with the control signal `if_source_s`.

10 Lines for the instruction set data element `if_ir_s`, the state word `id_state_s`, the instruction set data element `cs_ir_s` and the state element `cs_state_s` are connected to the inputs of multiplexer MUX2. A control input of the multiplexer MUX2 is provided with the control signal `if_source_s`.

 The multiplexer MUX3 is fed on the input side by the instruction set
15 data elements `if_ir_s` and `id_ir_s` and issues on the output side the instruction set data element `wb_ir_s`. The multiplexer MUX3 is controlled by the control signal `cs_ir_select_s`.

 The delay element D1 is fed on the input side by the context
identification value `context_id_i` and issues on the output side the context
20 identification value `cs_context_id_s`.

 The delay element D2 is fed by the useful data element `data_i` and
issues on the output side the useful data element `data_s`, with which the delay element
D3 is fed. The delay element D3 issues a useful data element `data_o`.

 A clock signal `clk_i` feeds the context switching unit CS, the instruction
25 fetch unit IF, the instruction decoding unit ID and the delay elements D1, D2 and D3.

The fundamental method of operation of the individual components of device 1 are described below.

The useful data elements `data_i`, whose processing is controlled by the device 1, originally come from different data sources and enter the device 1 in serial
5 fashion. Input of useful data elements `data_i` into the device 1 is laid out so that no more than one useful data element `data_i` is available for processing in one clock cycle.

Each data source is assigned to a thread, each thread having its own context. The context switching unit CS is notified by the context identification value
10 `context_id_i`, to which context the useful data element `data_i` being fetched for processing belongs.

The context switching unit CS contains a memory, in which the information concerning the context of each thread and the first instruction of each thread are entered. In addition, additional instructions are entered in the memory
15 registers of each context, which can be accessed by a forced jump. By the context identification value `context_id_i`, the context and first instruction of the thread, to which the useful data element `data_i` refers, is activated in the context switching unit CS.

From the context switching unit CS, by means of the instruction set
20 data element `cs_ir_s`, the instruction is transmitted to the instruction decoding unit ID, through which the processing step of the useful data element `data_i` being carried out is determined. In addition, the instruction decoding unit ID receives additional information through the state element `cs_state_s` that is entered in the corresponding registers of the context and refers to the actual state of the program prescribed for
25 running.

In order for the instruction set data element `cs_ir_s` and the state element `cs_state_s` to reach the instruction decoding unit ID, multiplexer MUX2 must be in logic state 0. For this purpose, the control signal `if_source_s` must be deactivated.

- 5 The context identification value `cx_context_id_s` also enters the instruction decoding unit ID, which corresponds to the context identification value `context_id_i`, delayed by one clock cycle.

- 10 The instruction transmitted by the instruction set data element `cs_ir_s` is decoded by the instruction decoding unit ID. The control signal `dec_o` is issued as a result of decoding by the instruction decoding unit ID. The control signal `dec_o` represents an output signal of device 1 and serves to control a unit that is connected after device 1, and in which the instructions processed by device 1 are to be executed.

- 15 In addition, the control signal `cs_store_s`, the speculative count value `id_spc_s`, the state value `id_state_s` and the instruction set data element `id_ir_s` are transmitted by the instruction decoding unit ID to the context switching unit CS. For transmission of the instruction set data element `id_ir_s`, the multiplexer MUX3 must be switched to the logic state 0. For this purpose, the control signal `cs_ir_select_s` is deactivated by the instruction decoding unit ID.

- 20 By means of the control signal `cs_store_s`, the contact switching unit CS is notified that data has been conveyed by the instruction decoding unit ID or by the instruction fetching unit IF to the contact switching unit CS and are to be stored there.

The speculative count value `id_spc_s` is taken up further below.

- 25 The state value `id_state_s` contains information on the state, in which processing of the thread at the output of the instruction decoding unit ID is situated.

The instruction set data element `id_ir_s` contains information, in order to overwrite the registers that contain the instruction set of the presently activated thread.

The instruction decoding unit ID also generates the control signal
5 `if_source_s`, with which the multiplexers MUX1 and MUX2 are controlled.

The instruction fetching unit IF obtains the speculative count value `cs_spc_s` and the context identification value `cs_context_id_s`, if the multiplexer MUX1 is in logic state 0. For this purpose, the control signal `if_source_s` must be deactivated.

10 The speculative count value `cs_spc_s` is taken up further below.

The instruction fetching unit IF contains a program memory. With reference to the speculative count value `cs_spc_s` and the context identification value `cs_context_id_s`, the instruction set is selected from the program memory, which is possibly required in the next clock cycle. This instruction set can be conveyed by
15 means of the instruction set data element `if_ir_s` in the form of a machine code to the context switching unit CS and to the instruction decoding unit ID. To convey the instruction set data element `if_ir_s` to the context switching unit CS, the control signal `cs_ir_select_s` must be activated, so that the multiplexer MUX3 is in logic state 1.

In addition, the context identification value `if_context_id_s` is issued by
20 the instruction fetching unit IF. The context identification value `if_context_id_s` corresponds to the context identification value `cs_context_id_s`, delayed by one clock cycle.

Cooperation of the individual components of device 1, and therefore the method of function device 1 are described in detail below.

The instruction structure underlying the instructions to be processed by device 1 reads:

repeat X until Y else go to Z (1)

5 X stands for an instruction, Y for a condition and Z for a static target information. In the present practical example of the device according to the invention, an instruction set has the three aforementioned data. The instruction sets are transferred by the instruction set data elements `cs_ir_s`, `id_ir_s` and `if_if_s`.

10 The instruction set to be processed first in a thread is entered in the registers of the corresponding context in the context switching unit CS. The context is activated by means of the context identification value `context_id_i`, and the instruction set entered in the registers is conveyed to the instruction decoding unit ID by means of the instruction set data element `cs_ir_s`. The instruction decoding unit ID decodes the instruction contained in the instruction set and generates the control signal `dec_o` from
15 it.

Parallel with the procedure just described, the useful data element `data_i` passes through the delay elements D2 and D3, during which, in each of the two delay elements D2 and D3, it is delayed by one clock cycle. Since activation of the context, as well as decoding of the corresponding instruction, requires the duration of
20 one clock cycle, it is ensured by delaying the data element in the delay elements D2 and D3 that the useful data element `data_i`, in the form of a time-delayed useful data element `data_o`, simultaneously reaches the downline component with the control signal `dec_o`, resulting from decoding of the corresponding instruction. In the downline component, processing of the useful data element `data_o` can then be carried
25 out with reference to control signal `dec_o`.

During the clock cycle, in which the instruction set belonging to the activated context is conveyed to the instruction decoding unit ID, the speculative count

value `cs_spc_s` is also transferred to the instruction fetching unit IF. The speculative count value `cs_spc_s` states that increment of the position of the instruction conveyed to the instruction decoding unit ID within the corresponding thread. If, for example, the first instruction of the thread is conveyed to the instruction decoding unit ID, the speculative count value `cs_spc_s` is equal to 2. By means of the speculative count value `cs_spc_s` and the context identification value `cs_context_id_s`, the instruction fetching unit determines in its program memory the instructions that correspond to the speculative count value `cs_spc_s`. In the aforementioned example, this would be the instruction set that contains the second instruction to be processed in the thread. By means of the instruction set data element `if_ir_s`, the determined instruction set can be sent to the context switching unit CS.

In order for the instruction set data `if_ir_s` to reach the context switching unit CS in the form of an instruction set data element `wb_ir_s` from the instruction fetching unit ID, the multiplexer MUX3 must be brought into the logic state 1 by activation of the control signal `cs_ir_select_s`.

Since the instruction structure of device 1 is based on a repetition loop according to the above instruction structure (1), it can happen that the instruction made available by the instruction set data element `if_ir_s` is not necessary, but that the previous instruction is carried out again. In this case, the instruction set determined by the instruction fetching unit IF is discarded.

During each repetition of instruction, the control signal `dec_o` is issued again by the instruction decoding unit ID. Repetition of an instruction is carried out, until the condition Y is met. Condition Y can be coupled to a wide variety of events.

During reperformance of an instruction, the control signal `cs_ir_select_s` is deactivated and the controls signal `cs_store_s` is activated. By deactivation of control signal `cs_ir_select_s`, the instruction set data element `id_ir_s` is passed through by the multiplexer MUX3 and reaches the context switching unit CS in

the form an instruction set data element `wb_ir_s`. The instruction set data element `wb_ir_s` contains the already executed and still to be repeated instruction. The control signal `cs_store_s` indicates to the context switching CS that it is to store the instruction set data element `id_ir_s`.

5 To control a repetition loop, there are a variety of possibilities. For example, the interrupt condition Y of a repetition loop can be stipulated by fulfillment of the equation “count + 0”. The value count is then contained in the instruction set and states the number of repetitions to be carried out at the beginning of the repetition loop. During each passage of the instruction set through the instruction decoding unit
10 ID, the value count is decremented. The instruction set updated by this is conveyed by means of the instruction set data element `id_ir_s` to the context switching unit CS and stored there, instead of the previous instruction set. As soon as the value count has reached the value 0, condition Y is fulfilled and the repetition loop is interrupted.

 As an alternative to the aforementioned example, the condition Y can
15 also be fulfilled by an external stipulation or by the input of a specific useful data element `data_i` or by the presence of a specific state of the program, which is reflected in the statement of the state value `cs_state_s` or `id_state_s`.

 As soon as the condition Y is met, processing of the thread jumps to the target information Z. This is consequently a conditioned jump instruction. The target
20 data Z always refers to instruction from the current thread.

 The condition Y is checked in the instruction decoding unit ID. All control signals are also adjusted from there.

 If the target Z concerns the subsequent instruction in the thread that is contained in the instruction set data `if_ir_s`, the control signal `if_source_s` is activated,
25 so that the instruction set data element `if_ir_s` is conveyed directly to the instruction decoding unit ID by the instruction fetching unit IF via multiplexer MUX2, and the

instruction contained in it can be decoded. Owing to the fact that the multiplexer MUX1 is in logic state 1, the speculative count value `id_spc_s` is transmitted to the instruction fetching unit IF. The speculative count value `id_spc_s` causes the instruction fetching unit IF to fetch the subsequent instruction in the subsequent clock cycle for the instruction just conveyed to the instruction decoding unit ID.

In addition, in this case, the control signals `cs_ir_select_s` and `cs_store_s` are activated, so that the instruction set stored in the registers of the corresponding context are overwritten with the instruction set fetched by the instruction set data element `if_ir_s`.

=f the target Z states a target address, whose corresponding instruction is situated in the context switching unit CS, this instruction is fetched by means of the state value `id_state_s` from the context switching unit CS and transmitted, by means of the instruction set data element `cs_ir_s` to the instruction decoding unit ID. For this purpose, control signal `if_source_s` must be deactivated.

To summarize, an algorithm is given below, which runs in the instruction decoding unit ID, and from which it follows, by means of which criteria the control signal `cs_ir_select_s`, `cs_store_s` and `if_source_s` are adjusted:

If (subsequent instruction of the thread is required), then

```
set cs_store_s to active
set cs_ir_select_s to active
if (cs_context_id_s ≠ if_context_id_s) then
    set if_source_s to inactive
else
    set if_source_s to active
end if
else
    set if_source_s to inactive
```

if (instruction set registers are overwritten), then

```
set cs_store_s to active
set cs_ir_select_s to inactive
else
    set cs_ir_select_s to don't care
    set cs_store_s to inactive
end if
end if
```

- 5 In the aforementioned algorithm, it should be kept in mind that the condition "instruction set registers are overwritten" is also true, if an instruction entered in the context switching unit CS is required.